VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Optimalizace pomoci kolonií mravenců

Ant Colony Optimization

2010

Jan Vašata

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava*.

V Ostravě 7. dubna 2010 .....

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7. dubna 2010 .....

Rád bych na tomto místě poděkoval všem, kteří mi pomohli s touto prací, především pak svému vedoucímu, Janu Platošovy, za jeho vedení při tvorbě práce, trpělivost a čas při konzultacích.

Abstrakt

ACO se zabývá řešením kombinatorických problémů, pomocí chování mravenců v přírodě. Přesněji na vyhledávaní nejkratší cesty v grafu pomocí feromonu. ACO jsou heuristické algoritmy. Nikdy se nám nezaručí, že získáme nejlepší možný výsledek. Tato technika se hodí například pro řešení problému obchodního cestujícího. Tato práce se zabývá implementaci ACO algoritmu.

Klíčová slova: ACO, TSP, optimalizace, bio-inspirované výpočty

Abstract

ACO deals with solving the combinatorial problem by using ants behavior in nature. More precisely, to search the shortest paths in graphs using pheromone. ACO algorithms are heuristic ones.We have never any guarantee that we get the best possible result. This technique is useful for example for solving the traveling-salesman problem. This work considers with the implementation of the ACO algorithm.

Keywords: ACO,TSP,optimization, bio-inspired computing

Seznam použitých zkratek a symbolů

TSP	–	Travelling salesman problem
ACO	–	Ant colony optimalization
AS	–	Ant system
VRP	–	Vehicle routing problem

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Ant colony optimalization (dále jen ACO) je algoritmus, který řeší například problém obchodního cestujícího. Tento algoritmus je založen na reálném chování mravenců v přírodě. Přesněji na jejich způsobu vyhledávání nejkratší cesty. K vytyčení nejlepší cesty se zde používají feromony, které pokládají na zem, po které mravenci putují. Množství jednotlivých feromonů na cestách poté rozhodují při výběru cesty dalšího mravence. Při dostatečném množství průchodu mravenců by nejkratší možná cesta měla mít největší množství feromonů. Algoritmus ACO vychází z této myšlenky. Hledá v grafu nejlepší cestu z bodu A do bodu B, za pomoci feromonu, který je ukládán na hrany grafu mravenci při jednotlivých průchodech grafu.

2 Základní myšlenka

2.1 Chování reálných mravenců

Mravenci jsou sociální hmyz, který žije v koloniích. Kolonie, česky bychom mohly říct mraveniště, obsahuje až miliony mravenců a nachází se na většině míst naší Země. Na světě se vyvinuly mnohé druhy mravenců, jejíž chování je specifické a přitahuje mnoho vědců ke zkoumání. Každý jednotlivý mravenec je začleněn do strukturované sociální sítě a má svou úlohu. Díky tomu můžou mravenci překonávat složité situace, které by vyžadovaly logické myšlení.

Mravenci k splnění úkolu mezi sebou využívají koordinaci, tím pádem neplní jeden úkol jeden mravenec, ale podílí se na něm vice mravenců. Příkladem může byt hledání cesty k potravě od mraveniště a zpět do mraveniště. Jelikož mravenci nemají rozvinuté některé orgány k vnímaní okolí, tak k procházení okolím používají feromony. Feromon je chemikálie, kterou můžeme chápat jako mravenčí pach, který dokážou mravenci rozpoznat. Takovýto přístup vnímaní světa mravenci byl experimentálně dokázán.

Chováni reálných mravenců, při hledáni potravy, můžeme zjednodušené popsat takto:

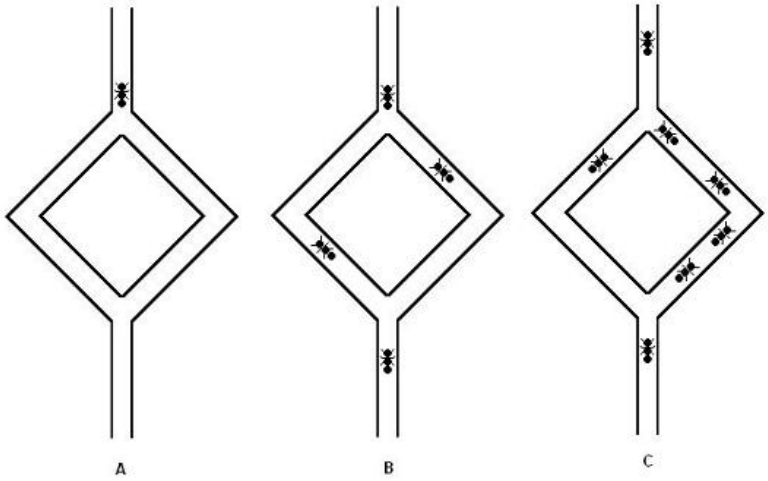
1. Mravenec vyjde z mraveniště za potravou.
2. První mravenec si vybírá cestu naprosto náhodně, protože se nemá podle čeho řídit.
3. Cestou pokládá na zem feromony.
4. Jakmile mravenec dorazí k potravě vrátí se do mraveniště.
5. Další mravenci, co vyjdou z mraveniště, se již řídí podle dříve položených feromonů. A i oni pokládají feromony na cestu po které jdou. A při výběru cesty mají tendenci vybírat cestu s největším množstvím feromonu.
6. Feromony se s uplynulým časem vypařují a proto zůstávají jen na cestách, které jsou vytížené.

Na obrázku 1 můžeme vidět tento postup.

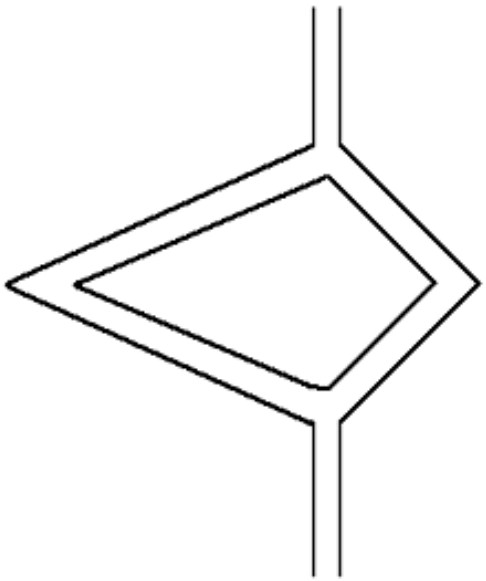
2.2 Experimenty s reálnými mravenci

Toto chování bylo vyvozeno z řad experimentů, které byly s mravenci provedeny. Popis některého takového experimentu uvedu teď. Mějme cestu jako je na obrázku 2. Cesta se děli na dvě části, které se na konci k sobě zbíhají a spojuji. Jedno z těchto ramen bude dvakrát delší než druhé. Tudíž nejkratší cesta je cesta po kratším rameni.

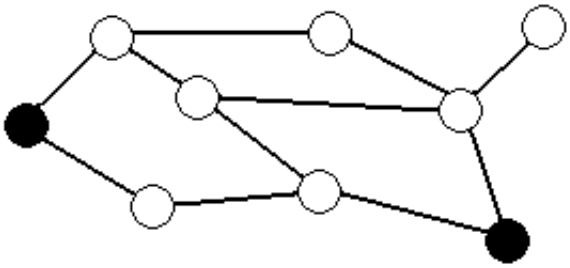
Nyní budou na tuto cestu vypuštěni mravenci. A jejich úkolem bude dostat se z jednoho konce cesty na druhý. První mravenec, jenž se dostane na tuto cestu, se zastaví u první křižovatky. Na křižovatce se bude rozhodovat, kterou cestu dál zvolit. Jelikož je první a před ním po této cestě nešel žádný mravenec, tak jeho rozhodnuti můžeme označit jako náhodné. Tudíž šance, že si zvolí kratší cestu, je stejná jako šance na delší



Obrázek 1: Chování mravenců při hledáni potravy. A) Mravenec náhodně nachází cestu k potravě. B) Další mravenci se již při výběru cesty rozhodují i podle feromonu na cestě. C) Většina mravenců při výběru cesty dávají přednost té, která má největší množství feromonů.



Obrázek 2: Cesta s dvěma odbočkami



Obrázek 3: Obecny graf

cestu. U druhé křižovatky předpokládejme pro jednoduchost, že mravenec půjde vždy správný směrem, tedy na konec cesty. Po průchodu cesty , na druhé straně mravence z cesty vyjmeme. Cestou mravenec pokládal na zem feromony, které se tam po určitou dobu uchovají, než se vytratí.

Na cestu budou nyní pokládáni další mravenci, kteří, až dojdou k první křižovatce, se budou rozhodovat, ale na cestě se již nacházejí feromony, takže jejich rozhodování nyní neovlivňuje jen náhoda, ale i již položené feromony. Toto rozhodováni je přímo úměrné velikosti feromonu na cestě. Nutno však zdůraznit, že jakkoliv velké množství feromonu nám nikdy nezaručí, že se daný mravenec touto cestou vydá. I když může byt šance na výběr cesty s největším množstvím feromonu velmi vysoká. Vždy se může nějaký mravenec odtrhnout a jít jinou cestou.

I další mravenci procházející cestou pokládají na zem feromony a zvyšuji tak její koncentraci. Feromony z cesty se neustále odpařují, z kratší cesty pomalu jak z delší cesty. takže se nám na kratší cestě koncentrace feromonu zvyšuje a na delší cestě zmenšuje. Po určitém množství průchodu většina mravenců už chodí kratší cestou, ale jednou za čas se i na delší trase nějaký mravenec objeví.

Tato technika nezaručí, že prvních par mravenců se k potravě dostane nejdřív, ale zaručí, že od dostatečného průchodu cestou se mravenci budou pohybovat nejkratší nalezenou cestou.

2.3 Od reálných mravenců k virtuálním

Když jsme pochopili, jak se reální mravenci chovají, tak nyní toto chování musíme převést na matematický problém.

Cestu si představíme jako graf o hranách (r,s), kde r je výchozí uzel a s je cílový uzel, které jsou spojeny hranou. Například takový jaký je na obrázku 3. Graf bude pro jed-

noduchost neorientovaný. Dále každá hrana (r,s) bude mít svou váhu $\tau(r,s)$, tedy číslo označující její cenu popřípadě velikost, bude to prvek, podle kterého se bude vypočítávat nejkratší cesta. V takovémto grafu budeme hledat nejkratší cestu ze startovního uzlu do cílového uzlu. Předpokládejme, že mravenec může projít daným uzlem pouze jednou. Protože při putování mravence jsou smyčky v grafu problémové, mají tendenci být atraktivnější čím víc mravenců prochází daným grafem, a celý algoritmus se přestává chovat, tak jak bychom očekávali. A pro výslednou nejkratší cestu se v cestě žádné smyčky vyskytovat nebudou.

Pravděpodobnost vyběru cesty na křižovatce označíme $p_k(r,s)$. Koncentraci feromonu na dané hraně označíme $\eta(r,s)$. Pak můžeme pro mravence vytvořit vzorec:

$$p_k(r,s) = \begin{cases} \frac{[\tau(r,s)]\cdot[\eta(r,s)]^\beta}{\sum_{u\in J_k(r)}[\tau(r,u)]\cdot[\eta(r,u)]^\beta} & jestlize \in J_k(r) \\ 0 & jinak \end{cases}$$

Kde β je parametr určující váhu mezi vzdáleností cesty a koncentraci feromonu na cestě. Hodnota β je vetší než 0. $\delta(r,s)$ je převrácena hodnota$\tau(r,s)$ ($\frac{1}{\delta(r,s)}$). Hodnota $\delta(r,s)$ velikost hrany.

Dále je potřeba definovat přidávání a vypařování feromonu z hran:

$$\tau(r,s) \to (1-\alpha)\cdot\tau(r,s) + \sum_{k=1}^{m}\Delta\tau_k(r,s)$$

kde

$$\Delta\tau_k(r,s) = \begin{cases} \frac{1}{L_k} & jestlize\ (r,s) \in nejlepsi\ nalezena\ cesta \\ 0 & jinak \end{cases}$$

a k značí mravence a m je celkový počet mravenců.

Přidávání a odpařování feromonu z hran přidělí vice feromonu na kratší cesty na všech hranách, které byly mravenci navštíveny, což vlastně způsobuje, že na kratších hranách je vetší koncentrace feromonu a jsou atraktivnější pro mravence při výběru cesty.

Výsledný algoritmus s použitím výše zmíněných vzorců by vypadal takto:

1. Přichystaní mravence v startovním uzlu.
2. Z uzlu aplikuj rozhodovací pravidlo.
3. Přesuň se na vybraný uzel. Zapamatuj si cestu.
4. Jestliže nejsi v cíli, přesuň se na (2).
5. Aplikuj pravidlo o odpaření a přidání feromonu.
6. Opakuj 1.

Smyčka v algoritmu by se měla provést tolikrát, kolikrát uznáme za vhodné. Při malém množství by ale algoritmus nemusel splnit účel.

3 ACO model a jeho využití

> Informatika jako věda o informacích a jejich zpracování chápe heuristiku jako postup získání řešení problému, které však není přesné a nemusí být nalezeno v krátkém čase. Slouží však nejčastěji jako metoda rychle poskytující dostatečné a dosti přesné řešení, které však nelze obecně dokázat. Nejčastější použití heuristického algoritmu nalezneme v případech, kde není možné použít jiného lepšího algoritmu, poskytujícího přesné řešení s obecným důkazem.[10]

ACO algoritmy jsou heuristické (Někdy též nazývané stochastické) algoritmy. Nikdy se nám nezaručí, že získáme nejlepší možný výsledek. V tom nejhorším případe, může vždy díky 'náhodě' mravenec zvolit pořád stejnou cestu jako první mravenec. Ani vysláním tisíců mravenců nemáme záruku, že daní mravenci se nevydají stejnou cestou. Proto výsledkem může byt i horší výsledek, než je možný. Avšak zvětšování množství vyslaných mravenců se dané riziko zmenšuje a možnost nalezení nejlepšího výsledku zvětšuje.

Myšlenkou využití modelu chování mravenců v diskrétní matematice se zabývalo již mnoho lidi, proto se ACO algoritmu vyvinulo mnoho. Každý algoritmus řešil nějaký známý problém, tabulka 1 nám ukazuje algoritmy, které vychází z ACO a byly použity na nějaký problém.

Vznikly algoritmy, které dokázaly vyhledávat v grafu cestu, řešit problém obchodního cestujícího nebo obarvovat grafy. Prvním ACO algoritmem byl Ant System (AS) vytvořením v roce 1991. Kdy se ukázalo, že takovýto systém dokázal vyřešit TSP. V roce 1997 pan Costa a pan Hertz navrhli AS-ATP algoritmus, který dokázal obarvit graf.

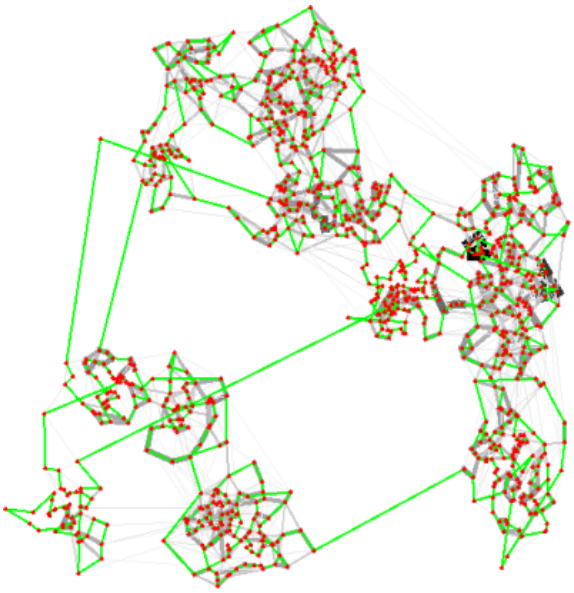
3.1 Obarvovani grafu

V roce 1997 navrhl Costa a Hertz AS-ATP systém pro přiřazovací problémy. Tento systém se lišil od standartního tím, že definoval na dvě feromonové cesty. Systém byl použít na problém obarvování grafu, kdy hledáme nejmenší počet barev, kterými by se dal graf obarvit. Při testování na 20 náhodných grafech o 100 uzlech se dosáhlo výsledku 15.05, nejlepší dosažený výsledek je 14.95.

3.2 Problém okružních jízd

Problém okružních jízd (VRP) je definován na grafu G=(U,H), U je množina uzlů a H je množina hran spojující tyto uzly. První uzel je označen za centrálni uzel (skladiště aut), ostatní uzly jsou uzly se zákazníky(místo odběru). Na uzlech se zákazniky je požadován odběr zboží. Zboží rozváží auta z cetrálního uzlu a po rozvozu jejich trať končí v cetrálním uzlu. Výsledkem je sestavení tras vozidel, tak aby byl každý zákazník právě jednou obsloužen.

V roce 1999 na tento problém navrhli Gambardella, Taillard a Agazzi system HAS-VRP, který vychází z ACO.



Obrázek 4: Výsledek TSP problému. [7]

Problem name	Authors	Year	Algorithm name
Traveling salesman	Dorigo, Maniezzo and Colorni	1991	AS
	Gambardella Dorigo	1995	Ant-Q
	Dorigo Gambardella	1996	ACS , ACS-3-opt
	Stutzle , Hoos	1997	MMAS
	Bullnheimer, Hartl and Strauss	1997	ASrank
Quadratic assignment	Maniezzo, Colorni Dorigo	1994	AS-QAP
	Gambardella, Taillard Dorigo	1997	HAS-QAPa
	Stutzle , Hoos	1998	MMAS-QAP
	Maniezzo , Colorni	1998	AS-QAPb
	Maniezzo	1998	ANTS-QAP
Job-shop	scheduling Colorni, Dorigo and Maniezzo	1994	AS-JSP
Vehicle routing	Bullnheimer, Hartl and Strauss	1996	AS-VRP
	Gambardella, Taillard and Agazzi	1999	HAS-VRP
Sequential ordering	Gambardella and Dorigo	1997	HAS-SOP
Graph coloring	Costa and Hertz	1997	ANTCOL
Shortest common	Michel and Middendorf	1998	AS-SCS

Tabulka 1: List of applications of ACO algorithms to static combinatorial optimization problems. Classification by application and chronologically ordered. [1]

3.3 Problém rozvrhování zakázkové výroby

Problém rozvrhování zakázkové výroby můžeme popsat takto: jsou zadané tři množiny. Množina úloh J, množina stroji M a množina operaci O. Každá úloha se skládá z operací. Cílem je vykonat vsechny operace všech úloh. Každá operace při provádění musí mít přiřazen nějaký stroj. Operace jednotlivých úloh musí následovat za sebou, tak jak jsou definovány. Žádná úloha nemůže být přerušena.

Colorni, Dorigo a Maniezzo v roce 1994 navrhli pro tuto úlohu Ant system (AS-JSP). Testování tohoto systému při zadání 15 strojů a 15 úloh vedlo k nalezení řešení do 10% optimální hodnoty. Výsledky tedy byly povzbudivé.

3.4 ACO a TSP

3.4.1 TSP

TSP je jeden z nejstarších NP-problémů. A jeho řešením se zabývalo již nespočet prací. Pro ACO algoritmy je TSP skvělý prostředek pro testování a ukázání, jak efektivní je ACO algoritmus.

> Problém obchodního cestujícího je obtížný diskrétní optimalizační problém, matematicky vyjadřující a zobecňující úlohu nalezení nejkratší možné cesty procházející všemi zadanými body na mapě.
>
> Laická formulace: Existuje n měst, mezi nimi silnice o známých délkách. Úkolem je najít nejkratší možnou trasu, procházející všemi městy a vracející se nazpět do výchozího města.
>
> Matematická formulace používající pojmosloví teorie grafů: Jak v daném ohodnoceném úplném grafu efektivně najít nejkratší hamiltonovskou kružnici? [8]

3.4.2 ACO pro TSP

Abychom mohli použít ACO model na TSP, tak musíme trochu poupravit algoritmus. Vstupem zde máme úplný graf, kde každá hrana je ohodnocena. Uzly jsou jednotlivá města, hrany jsou cesty mezi městy a ohodnocením hran se bude značit časová nebo vzdálenostní náročnost jednotlivých cest. Výsledkem musíme dostat Hemiltonovu kružnici. Změna oproti základnímu ACO, je taková, že daná trasa mravence, pro úspěšné zdolání musí zahrnovat všechny města. Tedy musí projít každým městem.

> V teorii grafů je hamiltonovská kružnice (také hamiltonovský cyklus) grafu taková kružnice v tomto grafu, která projde právě jednou všemi jeho vrcholy.[9]

Základní konstrukce algoritmu:

1. Vložíme do startovního uzlu mravence

2. Podle rozhodovacího pravidla vybereme další uzel (město), které musí mravenec navštívit.

3. Pokud zbývá nějaký nenavštívený uzel (město), přesun na (2)

4. Mravenec se vrátí do startovního uzlu.

Na daný graf budeme také používat metodu vypařovaní a přidávaní feromonu na hrany, jako je v základním modelu ACO.

4 Implementace

Pro konečnou implementaci ACO algoritmu jsem zvolil jazyk C++. V této kapitole rozeberu podrobněji implementovaný kód.

Byla vytvořena třída s názvem cACO, která v sobě obsahuje algoritmus pro výpočet. Jako vstup zde byl zvolen graf, jenž se skládá z hran, které jsou reprezentovány strukturou HRANA. Struktura Hrana je ještě obalena třídou cHranyV2, v třídě cACO pro lepší manipulaci. Třídě se dají nastavit, pro výpočet, startovní a cílový uzel, konstanty alfa(pro výpočet odpařovaní feromonu z hran) a beta(pro výpočet váhy feromonu při výběru cesty), a taky počet mravenců vyslaných v jednom cyklu(generaci) a počet těchto cyklu(generací). Byly vytvořeny dvě metody pro výpočet. Jedna metoda na hledáni nejkratší cesty v grafu a druha metoda na řešeni TSP problémů.

4.1 Struktura HRANY

Struktura vytvořena pro zástup hran v grafu s použitím ACO. Tato struktura je navržena tak, aby byla co nejjednodušší a aby odrážela požadavky na zpracování. Struktura obsahuje pozici, tedy počáteční a koncový bod hrany v grafu, která je reprezentována jako dvě celočíselné hodnoty v poli.

První hodnota je počátek hrany a druha hodnota je konec hrany, pro používání v orientovaném grafu. Jako další atribut je zde cena, která určuje, jak je daná hrana ohodnocena (atraktivní). Příkladem může byt délka hrany (vzdálenosti), časová náročnost a jiné. Cena je reprezentována celým číslem (integer). Posledním atributem je feromon. Atribut značí množství koncentrace feromonu na dané hraně. Je reprezentován jako číslo s pohyblivou řadovou čárkou (double).

```
struct HRANA
{
    int poz[2];
    int cena;
    double feromon;
};
```

Výpis 1: Struktura HRANY

4.2 Třída cHranyV2

Třída cHranyV2 obaluje strukturu HRANY pro lepší práci. Tato třída se chová jako obyčejné dynamické pole. Má metody pro přidávaní hran a modifikováni hran. Navíc je v této třídě implementovaná metoda pro odpařovaní feromonu z hran podle vzorce z kapitoly 2.3 , tato metoda je nazvaná PrepocetFer a jako parametr si bere číslo s pohyblivou řadovou čárkou, které určuje, jak rychle se mají feromony odpařovat z hran. Dále je implementována metoda pro přidávaní feromonu na vybrané hrany.

```
class cHranyV2
{
public:
    HRANA *hrany;
    int pocet;
    cHranyV2();
    void add(int a, int b, int c =1, double f = 1.0);
    void addModify(int a[2], double fer);
    void PrepocetFer(double fer);
    void clear();
};
```

Výpis 2: Třída cHranyV2

4.3 Třída cACO

Třída cACO slouží jako celek celého problému. Stará se o naplnění vstupními hodnotami, zpracování těchto hodnot a navrácení výsledku.

Nastavení této třídy se dá udělat pomoci těchto parametru:

1. start - Startovní uzel pro vyhledávaní
2. cil - cílový uzel pro vyhledávání
3. alfa - Míra vypařování feromonu z hran
4. beta - Váha feromonu na hranách vzhledem k ohodnocení hran
5. mapa - Graf, ve kterém budeme provádět výpočty
6. pocetGeneraci - počet vln, které budou vyslány
7. pocetMravencu - počet mravenců v jedné generaci

Třída obsahuje tyto metody:

1. Vypocet - slouží pro vyhledávaní v grafu
2. VypocetTSP - slouží pro řešení TSP problému
3. NastavGraf - slouží pro zadáni grafu
4. Cesty - slouží pro výběr cesty
5. SetPMravencu - nastavuje počet mravenců
6. SetPGeneraci - nastavuje počet generací
7. SetStart - nastavuje startovní uzel
8. SetCil - nastavuje cílový uzel

9. GetPMravencu - vrací počet mravenců
10. GetPGeneraci - vrací počet generací
11. GetStart - vrací startovní uzel
12. GetCil - vrací cílový uzel

```
class cACO
{
private:
    cHranyV2 mapa;

    int  start ;
    int  cil ;

    double alfa;
    double beta;

    int  pocetMravencu;
    int  pocetGeneraci;

    int  Cesty(int  i ,  cHrany historie ,  int  zaznam[255],double pr[255], int cena[255]);

public:
     cACO();

     void Vypocet();
     void VypocetTSP();

     void NastavGraf(int **uzly, int  pocet);


     void SetPMravencu(int i){pocetMravencu = i;};
     void SetPGeneraci(int i){pocetGeneraci = i;};

     void SetStart(int  i ){ start= i ;};
     void SetCil(int  i ){ cil  = i ;};

     int  GetPMravencu(){return pocetMravencu;};
     int  GetPGeneraci(){return pocetGeneraci;};

     int  GetStart(){return start ;};
     int  GetCil(){return  cil ;};
};
```

Výpis 3: Třída cACO

4.3.1 konstrukce algoritmu pro vyhledávání

Algoritmus pracuje s neorientovaným grafem, slouží pro vyhledání nejlepší cesty podle ohodnoceni hran v grafu. Nedovoluje vícenásobný průchod jedním uzlem. To proto, že při vyhledáváni by to způsobovalo v cestě smyčky, které jsou nežádoucí. Tento algoritmus se nehodí na řešeni problémů s TSP.

Tedy konečný algoritmus vypadá takto:

Cyklus generaci (vln)
- Cyklus jednoho mravence
  - Nastaveni mravence do startovního uzlu
  - Cykluspro cestu mravence
    - Zjištění poctu volných cest z uzlu
    - Jestliže je počet 0 zabit mravence
    - Vybrat cestu z uzlu
    - přesunout se do vybraného uzlu
    - Zapamatovat si předchozí uzel
    - Jestliže jsem v cíli
      - Porovnatnejlepší dosazené cesty s aktuální cestou
        - Zapamatování cesty
  - Odpařit z hran feromony
  - Přidat na hrany nejlepší cesty feromony

Rozhodovací pravidlo

Pravidlo pro rozhodování vychází ze vzorce z kapitoly 2.3. Pomocí β ovlivňujeme rozhodování, přesněji váhu mezi množstvím feromonu na hraně a cenou hrany.

```
double n = 1.0/CenaHrany;
m = feromon*pow(n,beta);
sance = m/ vm;
```

Výpis 4: Rozhodovací pravidlo

kde vm je součet všech m z možných cest pro výběr.

Aktualizační pravidlo

Pravidlo pro aktualizaci hran vychází z vzorce z kapitoly 2.3. Nejprve všechny hrany vynásobíme (1.0-α) kde α je konstanta pro určeni rychlosti vypařovaní feromonu. A následně přičteme k hranám nejlepší cesty feromony.

```
for(int i=0; i< pocetHran; i++)
{
    hrany[i].feromon = hrany[i].feromon * fer;
}
```

Výpis 5: algoritmus pro odpařování feromonu

```
for(int i = 0; i < nejCesta.pocet; i++)
{
        double fer = 1.0/delkaNejT;
        nejCesta.hrany[i].feromon += fer;
}
```

Výpis 6: algoritmus pro přidávaní feromonu

Generace mravenců

Mravence vysíláme ve vlnách, abychom zefektivnili práci. Je lepší provádět aktualizační pravidlo po určitém počtu průchodu, z kterých si vybereme nejlepší výsledek. Za předpokladu, že bychom nechtěli mravence posílat ve vlnách. Tak počet mravenců ve vlně nastavíme na 1 a k parametru GeneraceMravencu se chováme, jako kdyby to byl počet vyslaných mravenců.

4.3.2 Konstrukce algoritmu pro TSP

Konstrukce vychází z algoritmu z kapitoly 4.3.1. Pouze upravuje část, kdy se rozhoduje, kdy má daný mravenec ukončit svou pouť. Při TSP musí daný mravenec obejít všechny uzly. Tedy mravenec šťastně dorazil do cíle, když už neexistuje žádný další uzel, který nenavštívil.

Tedy konečný algoritmus vypadá takto:

Cyklus generaci (vln)
- Cyklus jednoho mravence
  - Nastaveni mravence do startovního uzlu
  - Cykluspro cestu mravence
    - Zjištění poctu volných cest z uzlu
    - Jestliže je počet 0 zabit mravence
    - Vybrat cestu z uzlu
    - Přesunout se do vybraného uzlu
    - Zapamatovat si předchozí uzel
    - Jestližejsou všechny uzly už navštívené
      - Porovnatnejlepší dosazené cesty s aktuální cestou
        - Zapamatování cesty
- Odpařit z hran feromony
- Přidat na hrany nejlepší cesty feromony

5 Testovaní

Vytvořený algoritmus na TSP problém byl otestován na třech příkladech. Tyto příklady byly vzaty ze stránek *ref odkaz*odkaz. Každý příklad je symetricky TSP problém. Stejné nastavení pro všechny příklady bylo $\alpha()$, $\beta()$ a počet mravenců v jedné generaci(10). Následně byla měřena průměrná procentuální odchylka od nejlepší cesty, která byly na tomto grafu nalezena. U každého příkladu uvádím kolik měl daný graf uzlů a nejkratší nalezenou trasu, která byla v tomto grafu nalezena. Při testování na graf postupně aplikujeme 1,10,50,100,150,200,250 a 300 generaci. Pro každý počet generaci několikrát najdeme nejkratší cestu a tu zprůměrujeme. Z výsledných hodnot vytvoříme graf.

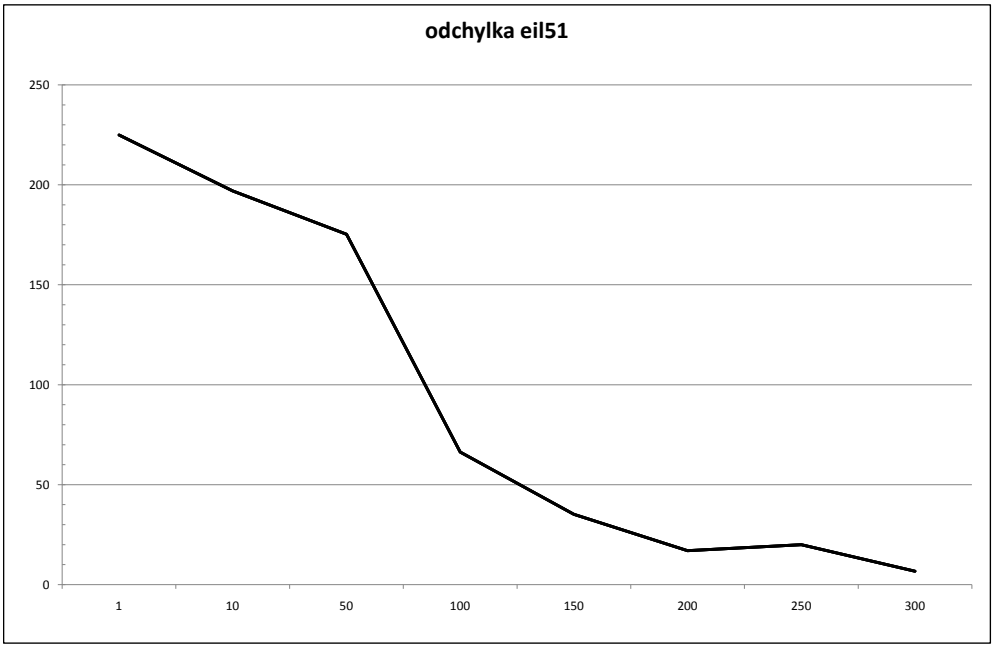
Příklad 1 (eil51.tsp)
Graf má 51 jedna uzlu.
Nejkratší nalezena cesta má délku 426.
Obrázek 5 ukazuje průběh testování algoritmu. Při vypuštění jen jedné generace je odchylka od nejkratší nalezené cesty víc jak 225%. Při vyslání 100 generaci už ale odchylka má 80%. Graf se od vyslání 50 až 150 generaci láme. A od 200 generaci je už odchylka od nalezené cesty pod 50% a od 300 vyslaných generaci je odchylka 8%. Tedy má nalezená cesta byla o 8% delší než nejlepší nalezena cesta v grafu.

Příklad 2 (eil76.tsp)
Graf má 76 jedna uzlu.
Nejkratší nalezena cesta má délku 538.
Obrázek 6 ukazuje průběh testování algoritmu. Při vypuštěni jen jedné generace je odchylka od nejkratší nalezené cesty víc jak 300%. Při vysláni 100 generaci už ale odchylka má 150%. Graf se od vyslání 50 až 200 generaci láme. A od 200 generaci je už odchylka od nalezené cesty pod 50% a od 300 vyslaných generaci je odchylka 47%. Tedy má nalezena cesta byla o 47% delší než nejlepší nalezena cesta v grafu.

Příklad 3 (eil101.tsp)
Graf má 101 jedna uzlu.
Nejkratší nalezená cesta má délku 629.
Obrázek 7 ukazuje průběh testování algoritmu. Při vypuštění jen jedné generace je odchylka od nejkratší nalezené cesty víc jak 350%. Při vysláni 100 generaci už ale odchylka má 200%. Graf se od vyslání 50 až 200 generaci láme. A od 200 generaci je už odchylka od nalezené cesty pod 75% a od 300 vyslaných generaci je odchylka 53%. Tedy má nalezena cesta byla o 53% delší než nejlepší nalezena cesta v grafu.

5.1 Souhrn

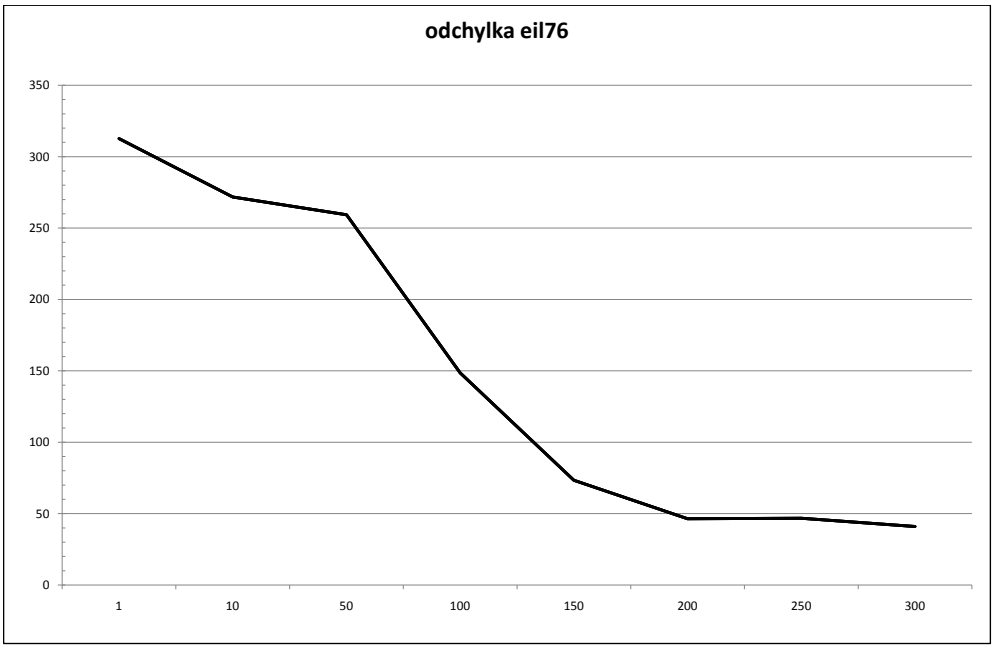
Otestovaní algoritmu nám ukázalo, že i když vyšleme minimum mravenců, dostaneme výsledek, který je ale velice nepřesný. Zvyšováním vyslaných mravenců roste šance na nalezení kratší cesty. Nejlepší výsledky jsou zobrazeny v tabulce 2.



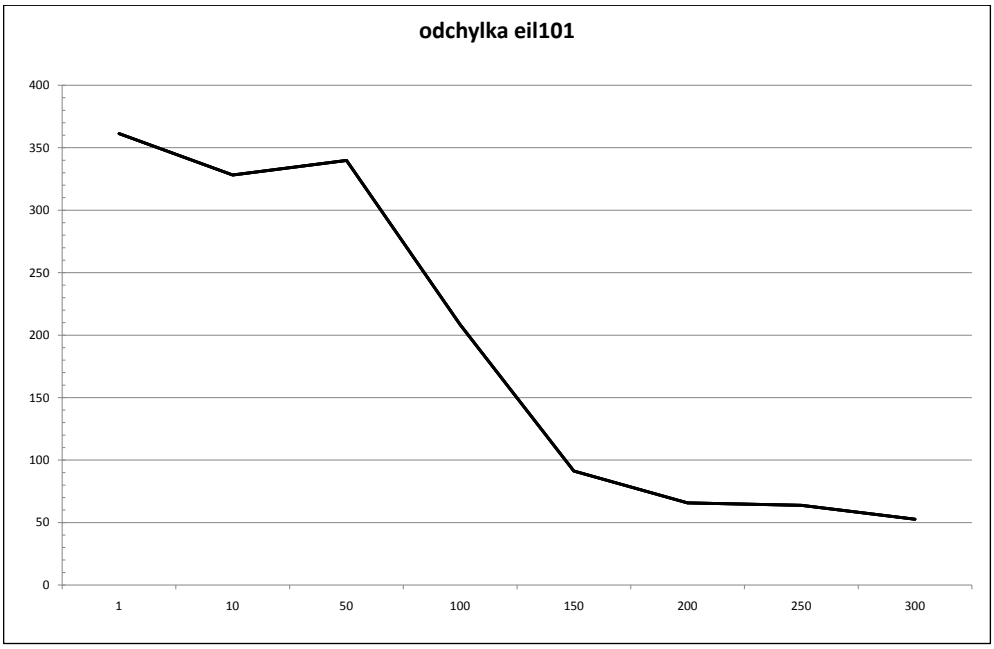
Obrázek 5: Graf pro eil51.tsp

soubor dat	Nej. cesta	počet uzlu	Nej. výsledek	α	β	počet generaci	% odchylka
eil51	426	51	448	0.1	0.5	300	5.2
eil76	538	76	580	0.1	1	300	7.8
eil101	629	101	662	0.1	2	300	5.2

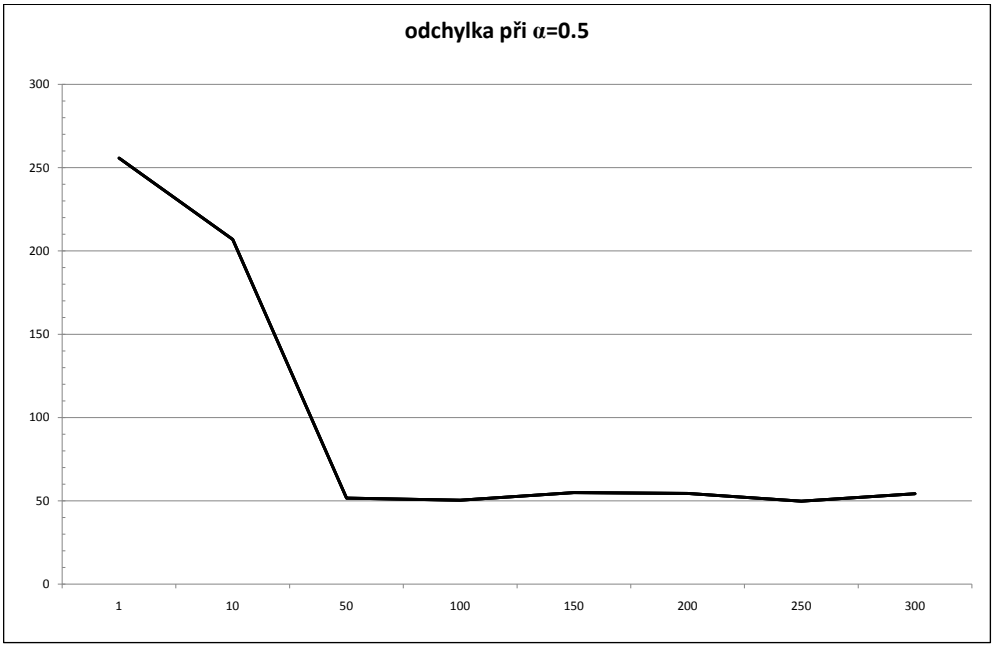
Tabulka 2: Porovnaní výsledku



Obrázek 6: Graf pro eil76.tsp



Obrázek 7: Graf pro eil101.tsp



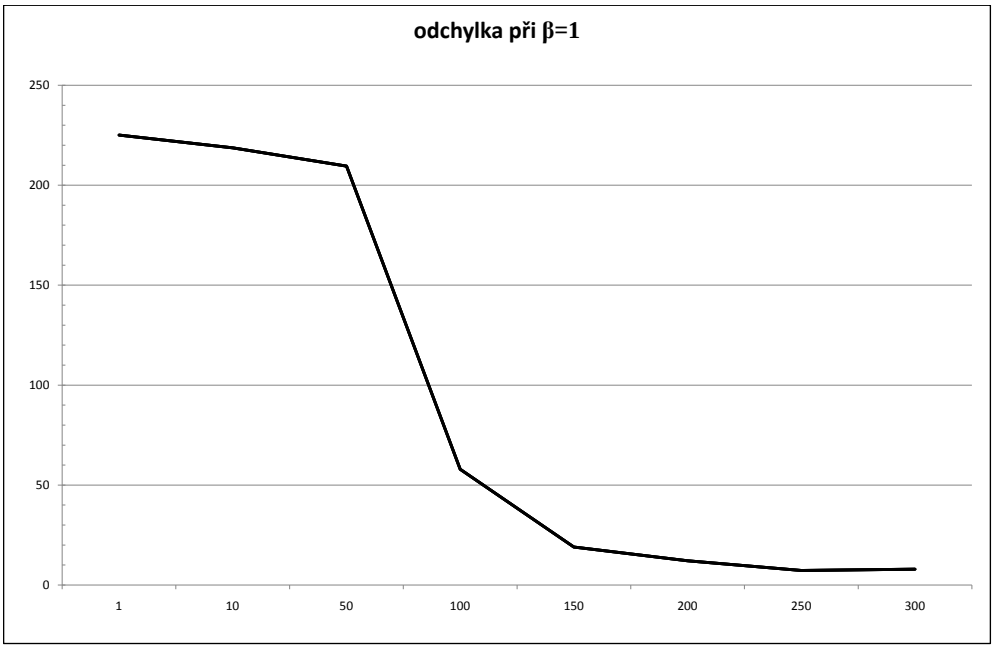
Obrázek 8: Graf pro eil51.tsp při alfa = 0.5

5.2 Závislost α,β

Jak závisí α a β na výsledné permutaci, bylo zjištěno testováním. Nejprve bylo α ponecháno konstantním na 0.1 a β byla nastavena z 0.5 na 1. A podruhé zůstala β konstantní na 0.5 a α byla nastavena z 0.1 na 0.5. Výsledné grafy si můžete prohlédnout na obrázku 8 a 9. Test byl proveden při poctu 10 mravenců na generaci na souboru dat z eil76.tsp při prvním pokusu a na eil51.tsp při druhém pokusu.

Jak je vidět, zvýšení β na 1 mělo za následek zlepšení výsledku testování. Už při 300 generaci máme vynikající výsledek.

Na druhou stranu zvýšení α nepříznivě ovlivnilo průběh grafu. Odpařování feromonů z hran bylo natolik rychlé, že se v daném grafu nemohly pořádně vytvořit cesty a tudíž to zhoršilo výsledky.



Obrázek 9: Graf pro eil76.tsp při beta = 1

6 Závěr

ACO je přírodní inspirace, která funguje velmi dobře. Heuristické algoritmy se hodí pro řešení některých specifických problému, díky své flexibilitě. I když nezaručují přesný a nejlepší výsledek, dokáží se k takovému výsledku přiblížit. Už dnes se tyto algoritmy používají a i nadále se používat budou. Výsledný na implementovaný kód se určitě nebude používat v praxi, při hledání řešení. Ale mohl by byt základem pro ty, co by jej tvořily.

Firma AntOptima [11] je pěknou ukázkou ACO algoritmu v praxi. Je to švédská firma, která využívá ACO pro reálné úkoly. Na svých stránkách nabízí řadu služeb (Vehicle routing, Data Mining, aj.), jenž řeší právě pomocí ACO.

7 Literatura

[1] Marco Dorigo,Thomas Stutzle, *Ant Colony Optimization*,1. vyd. London: The MIT Press, 2004. 301 s. ISBN 0-262-04219-3

[2] Marco Dorigo, Alberto Colorni, *The Ant System: Optimization by a colony of cooperating agents*,1. vyd. Brussels: IRIDIA, 1996. 24. s

[3] Marco Dorigo, Luca Maria Gambardella, *Ant Colony System:A Cooperative Learning Approach to the Traveling Salesman Problem*,1.vyd. Brussels: IRIDIA, 1997. 24. s.

[4] Marco Dorigo , Gianni Di Caro, *Ant Algorithms for Discrete Optimization*, 5. vyd. Brussels: IRIDIA, 1999. 37. s.

[5] Marco Dorigo , Krzysztof Socha, *An Introduction to Ant Colony Optimization*, 1. vyd. Brussels: IRIDIA, 2006. 19. s.

[6] Marco Dorigo, Mauro Birattari, Thomas Stutzle, *Ant Colony Optimization Artificial Ants as a Computational Intelligence Technique*, 1. vyd. Brussels: IRIDIA, 2006. 12. s.

[7] Kovářík, Skrbek, *Ant Colony Optimization with Parallel Subsolutions Heuristic. In European Simulation and Modelling Conference 2008*, Ghent: EUROSIS - ETI, 2008. 251. s.

[8] Problém obchodního cestujícího In Wikipedia : the free encyclopedia [online]. St. Petersburg (Florida) : Wikipedia Foundation, 22. 10. 2004, 27. 3. 2009 [cit. 2010-05-02]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Problém_obchodního_cestujícího>.

[9] Hamiltonovská kružnice In Wikipedia : the free encyclopedia [online]. St. Petersburg (Florida) : Wikipedia Foundation, 29. 3. 2005, 4. 12. 2008 [cit. 2010-05-02]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Hamiltonovská_kružnice>.

[10] Heuristické algoritmy In Wikipedia : the free encyclopedia [online]. St. Petersburg (Florida) : Wikipedia Foundation, 6. 2. 2008, 5. 4. 2010 [cit. 2010-05-02]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Heuristické_algoritmy>.

[11] Antoptima [online]. 2005 [cit. 2010-05-02]. Antoptima SA. Dostupné z WWW: <http://www.antoptima.com/site/en/index.php>.